CHAPTER 5
IMPLEMENTATION AND ANALYSIS

5.1 Implementation

In this process, data labeling is done by recognizing the train and test categories then converting the image into an array and doing a little image processing. resulting in X_train and Y_test in pickle form which will be saved in binary format using the pickle library which is used as input into the image extractor and into the algorithm.

```
1.  def Process_Image(self):
2.  X_Data = np.asarray(self.x_data) / (255.0)
3.  Y_Data = np.asarray(self.y_data)
4.  X_Data= X_Data.reshape(-1, self.IMAGE_SIZE,
self.IMAGE_SIZE,
5. return X_Data, Y_Data
6. def pickle_image(self):
7. X_Data,Y_Data = self.Process_Image()
8. pickle_out = open('X_Train','wb')
9. pickle.dump(X_Data, pickle_out)
10. pickle_out.close()
11. pickle_out = open('Y_Test', 'wb')
12. pickle.dump(Y_Data, pickle_out)
13. pickle_out.close()
14. return X_Data,Y_Data
```

On line 1, the function is to convert the image to array after recognizing the category based on Train and Test from the directory.

On line 2 and 3 make the image into an array value so that it can be processed by image estractor. Converting images to array values is done by using the Numpy library and normalizing it by dividing the image array value by 255.

On line 4 and 5, image processing is performed to resize the image and then returns the value to X_train and Y_test.

On line 6 to line 14, save the converted value into an array and name it according to the X_train and Y_test categories and save it in binary format using the pickle library for later use as input to the image extractor section.

5.1.2 Image exctraction & Train model

```
1. pickle_in = open("X_Data","rb")
2. X = pickle.load(pickle_in)
3.
4. pickle_in = open("Y_Data","rb")
5. y = pickle.load(pickle_in)
6. #di normalisasi lagi
7. X = X/255.0
8.
9. dense_layers = [0, 1, 2]
10.      layer_sizes = [16, 64, 128, 256, 384]
11.      layer_sizes2 = [128, 64, 32]
12.      conv_layers = [1, 2, 3]
13.      #memasukan data X utk dilakukan image Ekstraktor
14.      for dense_layer in dense_layers:
15.          for layer_size in layer_sizes:
16.              for conv_layer in conv_layers:
17.                  NAME = "{}-conv-{}-nodes-{}-dense-
   {}".format(conv_layer, layer_size, dense_layer,
   int(time.time()))
18.                  print(NAME)
19.
20.                  model = Sequential()
21.
22.                  model.add(Conv2D(layer_size, (3, 3),
   input_shape=X.shape[1:]))
23.                  model.add(Activation('relu'))
24.                  model.add(MaxPooling2D(pool_size=(2, 2)))
25.
26.                model.add(Conv2D(layer_size2, (3, 3),
   input_shape=X.shape))
27.                  model.add(Activation('relu'))
```

```
28.                     model.add(MaxPooling2D(pool_size=(2, 2)))
29.
30.                     for l in range(conv_layer-1):
31.                         model.add(Conv2D(layer_size2, (3, 3)))
32.                         model.add(Activation('relu'))
33.                         model.add(MaxPooling2D(pool_size=(2,
   2)))
34.
35.                     model.add(Flatten())
36.
37.                     for _ in range(dense_layer):
38.                         model.add(Dense(layer_size))
39.                         model.add(Activation('relu'))
40.
41.                     model.add(Dense(1))
42.
43.                     model.add(Activation('sigmoid'))
44.         model.compile(loss='binary_crossentropy',
45.                                 optimizer='adam',
46.                                 metrics=['accuracy'],
47.                                 )
48.
49.                     model.fit(X, y,
50.                               batch_size=32,
51.                               epochs=10,
52.                               validation_split=0.3,
53.                               callbacks=[tensorboard])
54.
55.       model.save('East_text4.model')
```

On line 1 to line 7, pre-processing the extractor image by loading the labeling results in the form of X_train and Y_test and normalizing it by dividing by the value 255.

On line 9 to line 12 define the number of layer size, dense layer and convolutional layer based on the EAST algorithm. There are 3 dense layers, the

number of layer sizes has a size of 16, 64, 128, 256, 384, layer size 2 has a size of 128, 64, 32, with a convolutional layer of 3.

On line 14 to line 28 perform a looping function for the dense layer and layer size with a convolutional layer model using a 3x3 kernel, using the activation function, namely ReLU, then using maxpooling to find important information in the image with a 2x2 kernel, then do the convolutional layer again with convolutional layer_size2 with a 3x3 kernel and use ReLU activation again and do the maxpooling process again with a 2x2 kernel, as a image extactor Feature Extractor Stem.

On line 30 to line 33 as an implementation of the Branch merging feature by doing a convolutional layer using the layer_size2 size (128, 64, 32) with a 3x3 kernel using the ReLU function activation and maxpooling with a 3x3 kernel size.

On line 35, flatten the data for merging data from the Feature Extraktor Stem and Feature Merging Branch processes.

On line 37 to line 43 perform a looping function for the dense layer to produce output before the train is carried out, then the output is carried out by activating the Sigmoid function.

On line 44 to line 47, process data with a compile model before entering the train by calculating loss using binary_crossentropy and optimizing the train using Adam Optimizer and calculating accuracy.

On line 49 to line 55, perform the Train model process using Batch size 32, and perform 10 Epoch Train steps with a validation of 0.3. Then the model will be saved in Tensorflow.keras format with the name East_text4.model.

5.1.3 Text Recognation

```
net = cv2.dnn.readNet("East_text4.pb")
def text_detector(image):
      orig = image
      (H, W) = image.shape[:2]

      (newW, newH) = (320, 320)
      rW = W / float(newW)
      rH = H / float(newH)

            image = cv2.resize(image, (newW, newH))
            (H, W) = image.shape[:2]

            layerNames = [
                  "feature_fusion/Conv_7/Sigmoid",
                  "feature_fusion/concat_3"]


            blob = cv2.dnn.blobFromImage(image, 1.0, (W, H),
                  (123.68, 116.78, 103.94), swapRB=True,
crop=False)

            net.setInput(blob)
            (scores, geometry) = net.forward(layerNames)

            (numRows, numCols) = scores.shape[2:4]
            rects = []
            confidences = []
            for y in range(0, numRows):
                  scoresData = scores[0, 0, y]
                  xData0 = geometry[0, 0, y]
                  xData1 = geometry[0, 1, y]
                  xData2 = geometry[0, 2, y]
                  xData3 = geometry[0, 3, y]
                  anglesData = geometry[0, 4, y]
            boxes = non_max_suppression(np.array(rects),
probs=confidences)
```

```
35.            for (startX, startY, endX, endY) in boxes:
36.                  startX = int(startX * rW)
37.                  startY = int(startY * rH)
38.                  endX = int(endX * rW)
39.                  endY = int(endY * rH)
40.                  boundary = 2
41.
42.                  text = orig[startY-boundary:endY+boundary,
   startX - boundary:endX + boundary]
43.                  text = cv2.cvtColor(text.astype(np.uint8),
   cv2.COLOR_BGR2GRAY)
44.                  textRecongized =
   pytesseract.image_to_string(text)
45.                  print(textRecongized)
46.                  cv2.rectangle(orig, (startX, startY),
   (endX, endY), (0, 255, 0), 3)
47.                  orig = cv2.putText(orig, textRecongized,
   (endX,endY+5), cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255),
   2, cv2.LINE_AA)
48.            return orig
49.
50.      unnamed =
   cv2.imread('/home/kevinew/Documents/East_project/panin.jpeg'
   )
51.
52.      array = [unnamed]
53.
54.      for i in range(0,2):
55.            for img in array:
56.                  imageO = cv2.resize(img, (640,320),
   interpolation = cv2.INTER_AREA)
57.                  orig = cv2.resize(img, (640,320),
   interpolation = cv2.INTER_AREA)
58.                  orig_gray = cv2.cvtColor(orig,
   cv2.COLOR_BGR2GRAY)
59.                  textDetected = text_detector(imageO)
60.                  textDetected_gray =
   cv2.cvtColor(textDetected, cv2.COLOR_BGR2GRAY)
61.                  cv2.imshow("Orig Image",orig_gray)
62.                  cv2.imshow("Text Detection",
   textDetected_gray)
63.                  time.sleep(2)
64.                  k = cv2.waitKey(30)
```

```
65.                     if k == 27:
66.                         break
67.     cv2.destroyAllWindows()
```

On line 1, load the model train from the prepared model train.

On line 2, make a function to predict the text area.

On line 4 to line 11 perform the Blob method to create a bounding box on the text area detected in the image based on the height and width of the image.

On lines 13 to line 15 carry out the Sigmoid Fusion Feature process and the Fusion concat Feature which is responsible for loading the convolutional layer model from model train to perform Feature Extraction stem on the input image and perform concat for merging stem extraction feature image and branch merging feature image from the train model.

On line 18 to line 40, mark the text area in the image using the Blob. Then give the location the geometry of the text area with a Blob. loop to perform geometry at Y coordinate based on probability at pixel value. Then perform a loop to calculate based on the X coordinates and Y coordinates along with the height and width, to create a text area marker box.

On line 42 to line 48, create a bounding box in the text area, perform image processing to make the image grayscale, perform text image extraction to recognize text. Displays text on the input image in the text area.

On line 50 it inputs an image with imread from the CV2 library.

On lines 54 to 62, resize the image then process the image by making the image grayscale, then display the grayscale result with the CV2 imshow, then display the original image with the CV2 imshow, display the detection text results along with the bounding box and text of the detection results on the image.

On lines 63 to 67 displays an image with CV2 waitkey with a time limit.

5.2 Analysis

5.2.1 Accuracy, Precission, Recall and F1-Score Formula

Akurasi = (TP + TN) / (TP+FP+FN+TN)

Precission = (TP) / (TP+FP)

Recall = (TP) / (TP + FN)

F1 Score = 2 * (Recall*Precission) / (Recall + Precission)

TP = True Positive

TN = True Negative

FP = False Positive

FN = False Negative

5.2.2 Table Accuracy, Precission and Recall

No	Character	Detected	Accuracy	Precission	Recall	F1-Score
1.	Paninkb	7	55,6 %	71,5%	71,5%	52,5%
2.	Matahari	8	100%	100%	100%	100%
3.	Tes	3	100%	100%	100%	100%
4.	Rafpless	8	71,5%	75%	75%	73,17%
5.	Paraggn	7	85,8%	85,8%	100%	92,30%

Table 5.1: Analysis

5.2.3 Analysis Table of Accuracy, Precission and Recall

In calculating the performance measurement, there are 4 types that have different standards, such measurements are measured through the level of accuracy, measurement using a precision and recall system, and calculation of the F1-score. Accuracy itself is the ratio of correct predictions in the form of positive and negative with the overall data, whereas Precission is the ratio of true positive predictions compared to the overall positive predicted results and Recall or Sensitivity Is the ratio of positive true predictions compared to the overall correct positive data, while the F1 Score is a comparison weighted average precision and recall. From the calculation data that has been applied using the accuracy formula, we get the result in the form of what percentage of the accuracy of an article that is classified by character and processed in the EAST Algorithm. The reason for using the accuracy measurement system is to compare the results obtained from my model with the EAST OCR model, so that we can determine the accuracy of the results of each model efficiently and quickly. The results obtained in the calculation using the accuracy formula are in the form of True Positive (TP) which is positive data that is detected correctly, True Negative (TN) is the number of negative data detected correctly, while False Positive (FP) is negative data but detected as positive data, and False Negative (FN), which is the number of negative data but is classified incorrectly by the system.